View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Calhoun, Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications | Faculty and Researcher Publications Collection

2016-12

Decomposition of MAC Address Structure for Granular Device Inference

Martin, Jeremy

http://hdl.handle.net/10945/50648

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

Decomposition of MAC Address Structure for Granular Device Inference

Jeremy Martin
United States Naval Academy
jmartin@usna.edu

Erik Rye
United States Naval Academy
rye@usna.edu

Robert Beverly
Naval Postgraduate School
rbeverly@nps.edu

ABSTRACT

Common among the wide variety of ubiquitous networked devices in modern use is wireless 802.11 connectivity. The MAC addresses of these devices are visible to a passive adversary, thereby presenting security and privacy threats – even when link or application-layer encryption is employed. While it is well-known that the most significant three bytes of a MAC address, the OUI, coarsely identify a device's manufacturer, we seek to better understand the ways in which the remaining low-order bytes are allocated in practice. From a collection of more than two billion 802.11 frames observed in the wild, we extract device and model information details for over 285K devices, as leaked by various management frames and discovery protocols. From this rich dataset, we characterize overall device populations and densities, vendor address allocation policies and utilization, OUI sharing among manufacturers, discover unique models occurring in multiple OUIs, and map contiguous address blocks to specific devices. Our mapping thus permits fine-grained device type and model *predictions* for unknown devices solely on the basis of their MAC address. We validate our inferences on both ground-truth data and a third-party dataset, where we obtain high accuracy. Our results empirically demonstrate the extant structure of the low-order MAC bytes due to manufacturer's sequential allocation policies, and the security and privacy concerns therein.

1. INTRODUCTION

802.11 wireless protocols are used in almost all commodity network devices, including a variety of appliances, sensors, and embedded systems. This Internet of Things (IoT) includes mobile phones, media players, access points, printers, cameras, thermostats, and automobiles, and is predicted to number 50 billion devices by 2020 [10].

Securing this vast footprint of wireless devices has thus taken on increased importance [28]. In this paper, we re-examine a specific weakness stemming from the use of 802.11: exposure of link-layer Media Access Control (MAC) addresses. A passive adversary within radio range can capture MAC addresses – a persistent globally unique identifier – even when link and application-layer encryption is employed. In addition to tracking and threats to user privacy, MAC addresses advertise coarse information about device manufacturers via the three most significant MAC bytes (the Organizationally Unique Identifier (OUI)) [13].

In this work, we seek to better understand the extent to which the remaining three *low-order* bytes can reveal information about an observed device. Our intuition is simple: if manufacturers allocate MAC addresses for a given OUI in a predictable way, i.e. sequentially from a subprefix range of the 2^{24} space, this information can more finely fingerprint the device. As a concrete example, within the `24:A2:E1` OUI block assigned to Apple, we find that approximately half of the space is allocated to iPhone 5c (GSM) models, while ∼20% is dedicated to the iPad Mini 2 (Cellular), and another ∼20% is for the iPad Mini 2 (WiFi). Crucially, the allocations are *contiguous blocks*. Thus, this vendor policy permits inference of specific device type and model information solely from the observed MAC address.

Associating MAC addresses with specific device models presents a dilemma: given only public capture data, how can we determine device model information? We build this mapping from a collection of over two billion frames observed in the wild, where we extract information leaked by 802.11 management frames (probe requests, probe responses, and beacons) and multicast Domain Name System (mDNS) packets. Our collection is entirely passive and we follow a strict IRB policy (detailed in §3.1).

While allocation practices vary across vendors, with several exhibiting significant complexity, we show that they are generally non-random. This determinism illustrates two concerns: i) management protocols allow significant privacy and security leaks; and ii) the structure and allocation of MAC addresses lends itself to device fingerprinting. Granular fingerprinting of wireless devices is valuable for supporting policy-based security and research efforts including crowd density [16] and population diversity [19]. Device fingerprinting permits profiling of user activity, habits, and movements – an activity currently performed in the retail sector [26, 25]. Commercial products such as [18] are designed for vehicle and pedestrian traffic monitoring, and rely on identifying device manufacturers.

More nefariously, fingerprinting can be employed to perform targeted attacks against a device [17, 7]. Recent work creates denial of service attacks against Google Glass [22], while commercial products can deauthenticate particular IoT

This paper is authored by an employee(s) of the United States Government and is in the public domain. Non-exclusive copying or redistribution is allowed, provided that the article citation is given and the authors and agency are clearly identified as its source.

ACSAC '16, December 05 - 09, 2016, Los Angeles, CA, USA

ACM ISBN 978-1-4503-4771-6/16/12…$15.00

DOI: http://dx.doi.org/10.1145/2991079.2991098

Table 1: Example Management Frame (WPS) Fields

Manufact	Model Name	Model Num	Device Name
Asus	Nexus 7	Nexus 7	razor
ASUSTeK	WPS Router	RT-N65R	ASUS WPS Router
Casio	C811 4G	C811 4G	Peer Device
Cisco	Linksys E2500	v1.0.05	Linksys E2500
HP	DeskJet 3630	3632	HP-DeskJet 3630
HTC	HTC One X	HTC One X	cingular_us
Huawei	Prism II	Prism II	U8686_TMO
LGE	LG-D415	LG-D415	w7_tmo_us
Samsung	SCH-R740C	SCH-R740C	amazing3gcri

devices (cameras, drones, etc.) [23]. Fine-grained fingerprinting allows a criminal to easily and cheaply perform reconnaissance, identifying e.g., 802.11-enabled security cameras, thermostats, and home security systems. Even automobiles have been targeted via 802.11 radios using fingerprinting techniques to isolate Uconnect devices [9]. More generally, the ability to systematically fingerprint and identify public works infrastructure such as water sensors, biometric and medical monitors, and industrial control systems, may expose these devices to targeted attacks.

We take a first step in decomposing MAC address structure to both provide a more granular MAC-based fingerprinting technique, as well as highlight security and privacy concerns from these ubiquitous wireless device identifiers. This paper thus makes the following primary contributions:

- Curation of a large corpus of 802.11 wireless data that captures a non-trivial cross-section of devices. From this data, we develop a technique to predict the specific model of a device from its MAC address, enabling finer-grained fingerprinting than previously possible.
- Analysis of the various MAC address allocation strategies employed by different vendors. We find that MAC addresses are assigned to devices in a nonrandom manner, with contiguous ranges of the OUI dedicated to distinct models. The structured allocation of MAC addresses lends itself to the development of a model fingerprinting strategy.
- Observation of OUI complexity, including multiple manufacturers using a single OUI (particularly IoT devices with third-party chipsets). Conversely, models commonly span multiple OUIs.
- Validation of our model fingerprinting against a third-party dataset with $> 80\%$ accuracy, and a practical application of locally assigned MAC address derandomization introduced in [6, 27]. Additionally, we achieve high precision and recall validating against a known set of 279 devices. Finally, we submit our corpus to a 5-fold cross-validation and observe $\sim$90% accuracy rate in predicting device models based solely on MAC addresses alone.

The remainder of this paper is organized as follows. We first present background in §2. Section §3 details our methodology, while §4 provides our analysis and results. We conclude in §5 with suggestions for future work.

2. BACKGROUND

2.1 MAC Addresses

Every network interface on an 802.11 capable device has a MAC address layer-2 hardware identifier. MAC addresses are designed to be persistent and globally unique. OUIs are purchased and registered with the Institute of Electrical and Electronics Engineers (IEEE) [13]; the manufacturer is then free to assign the remaining low-order three bytes (2^{24} distinct addresses) to devices. To the best of our knowledge, our work is the first to shed light on the allocation policies employed by manufacturers for assigning addresses *within and across* OUI blocks.

Table 2: Example mDNS Fields

OS	**key**	**value**	**Derived Model**
iOS	model	n27aap	Apple Watch (38mm)
iOS	model	k93ap	iPad 2 (WiFi)
iOS	model	n71map	iPhone 6s (TSMC A9)
iOS	model	k66ap	Apple TV 2G
BlkBry	hwid	0x9600240a	BlackBerry Cafe
BlkBry	hwid	0x04002607	BlackBerry Z10
Android	n	SAMSUNG-SM-N910A	Galaxy Note 4
Android	n	A0001_a1f0 (OnePlus)	OnePlus One

In addition to the public, globally unique manufacturer assigned MAC address, modern devices frequently use "locally assigned" addresses [5] which are distinguished by a Universal/Local bit in the most significant byte. Locally assigned addresses are used in a variety of contexts including access point (AP) Virtual Local Area Networks (VLANs), tethered hotspots, and peer-to-peer (P2P). For example, smart televisions connect to a wireless AP using a public MAC address, but also offer P2P connectivity, e.g., "WiFi-Direct" or "WiFi-Display." For these P2P connections and other types of hotspots, a locally assigned MAC is typically derived from the device's global MAC address at manufacturer-standardized interval offsets from the device's global MAC address.

Locally assigned addresses are also used to create randomized addresses as an additional measure of privacy. By using randomized, locally assigned MAC addresses that change over time, tracking a wireless device is no longer trivial. For this reason, we frequently observe 802.11 probe requests that use locally assigned addresses if the device is not associated with a known AP. However, as shown in [27] and §4.3, a device's WPS UUID-E can often be used to derive its global MAC from the randomized MAC address.

2.2 802.11 Management Frames

802.11 management frames are unencrypted and generated by devices automatically to discover networks and capabilities, associate or authenticate, and disconnect. 802.11 management data provide the crux of our device fingerprinting capability; we collect the following subtypes: *probe requests:* frames sent by a client to discover available APs and capabilities; *probe responses:* frames sent by an AP in response to a probe request; and *beacons:* periodic frames sent by APs to advertise presence and capabilities.

These frames are extensible, and many modern devices use them to support WPS. In addition to its intended purpose of wireless authentication, WPS often reveals model-specific information about a device, e.g. "Asus Nexus 7."

A recent study found up to 8.6% of client devices broadcast WPS fields [27]. We also consider WPS field data from AP devices, through probe responses and beacons. WiFi Alliance P2P standards such as WiFi-Direct and WiFi-Display are derivatives of WPS and mandate its use. Table 1 shows example WPS data from a small subset of devices.

2.3 Multicast DNS

A significant subset of 802.11 devices do not utilize WPS. When WPS is not available, we attempt to leverage the mDNS protocol. Wireless devices use mDNS for Domain Name System-Based Service Discovery (DNS-SD) as part of

a suite of zero configuration network protocols to advertise network services and capabilities [24] (e.g. Apple's AirPlay, AirPrint, and AirDrop services).

Many mDNS messages contain DNS-SD key-value pairs [24] that uniquely identify the device model of the source. For instance, as of iOS 8.0, Apple devices send mDNS messages that contain the `dns.txt` key of `model=` and a corresponding model identification value, e.g. `model=N61AP` corresponds to an iPhone 6. Blackberry devices encode model information using key `hwid=`, while several Android manufacturers use key `n=`. To resolve these model identification strings to common names, we utilize a variety of public online resources [14, 21]. Table 2 gives examples of observed manufacturer-specific implementations.

2.4 Related Work

It is well known that hardware identifiers reveal basic manufacturer details valuable to device fingerprinting [13, 20]; however, based solely on the MAC address structure it has not been possible to resolve beyond a device's manufacturer. To date, resolution of MAC addresses to device model granularity has required analysis of higher level protocols [20] or resource intensive analysis [11, 8].

In [20, 15] methodologies using higher layer application protocols for device fingerprinting in support of 802.11 and Global System for Mobile Communications (GSM) cross-correlation analysis are evaluated. The authors of [20] highlight several important issues counter to our goals: i) application layer protocols can be encrypted; ii) often require the device's user to perform some action in order to initiate the protocol transmission, reducing probability of passive detection; and iii) model granularity is rarely possible, specifically with iOS devices. Additionally, the techniques that rely on HTTP User-Agent strings or hostnames are prone to inaccuracy, due to the ease with which a user can obfuscate the fingerprint of the device [1, 3, 4]. Fingerbank [2], a community-sourced project, attempts to uncover the manufacturer and model of networked devices based on a combination of HTTP User-Agents, the order of DHCP option fields, and the OUI vendor registered with the IEEE. Altering the fields of interest for our primary methods, fingerprinting WPS and mDNS, would require extensive re-engineering of the kernel.

The authors of [11] and [8] illustrate the ability to fingerprint a mobile device's 802.11 device driver and Operating System (OS) using a timing-based analysis of 802.11 management frame probe requests. This fingerprinting method requires a steady stream of packet data from each device. Relying on a consistent stream of data can be problematic – in the case of vehicle and pedestrian congestion analysis a single packet can be the sole source of information. Our work improves on existing methods by introducing fingerprinting techniques that rely on a single transmitted frame, regardless of type or encryption method.

Hupperich et al. highlighted the growing need and inherent difficulties in creating a mobile device fingerprinting capability, and provided a solution involving a set of browser, OS, hardware, and user behavioral attributes [12]. These identifiers require the ability to collect the desired attributes as part of user web activity. Our work requires no such active process, relying solely on passive techniques. Furthermore, as previously mentioned, it is difficult to modify the data fields used by our fingerprinting methods.

Algorithm 1 Frame Processing Strategy

```
if frame = mgmt && mgmt_ext = wps then
    if src_mac ≠ universal then
        src_mac ← lookup(wps.uuid_e)
    src_mac, wps.* → database
    if frame = WiFi-Direct || WiFi-Display then
        src_mac, p2p.* → database
else if frame = mdns then
    model ← lookup(mdns.boardid)
    src_mac, mdns.*, model → database
if frame = beacon then
    src_mac, ssid → database
    if mgt_tag = apple && apple_type = 6 then
        src_mac, apple.*, ssid → database
```

3. METHODOLOGY

Over the course of approximately one year, we capture unencrypted 802.11 device traffic using inexpensive commodity hardware and open-source software. We primarily use an LG Nexus 5 Android phone running Kismet *PcapCapture* paired with an AWUS036H 802.11b/g Alfa card. We hop between the 2.4GHz channels 1, 6, and 11 to maximize coverage. We additionally employ several Raspberry Pi devices running Kismet with individual wireless cards each dedicated to channels 1, 6, and 11. Our corpus spans January 2015 to May 2016 and encompasses approximately 9,000 individual packet captures. The collection contains over 600 gigabytes (GBs) of 802.11 traffic, consisting of over 2.8 million unique devices across a spectrum of IoT devices.

3.1 Ethical Considerations

Our collection methodology is entirely passive. At no time did we attempt to decrypt any data, or perform active actions to stimulate or alter normal network behavior. Our intent is to show the ease with which one can build a similar capability to that in this paper with low-cost off-the-shelf equipment. However, given the nature of our data collection, we consulted with our Institutional Review Board (IRB).

The primary concerns of the IRB centered on: i) the information collected; and ii) whether the experiment collects data "about whom" or "about what." Because we limit our collection to 802.11 management frames, mDNS, and other layer-2 discovery protocols, we do not observe Personally Identifiable Information (PII). Although we observe IP addresses, our experiment does not use these layer-3 addresses. Even with an IP address, we have no reasonable way to map the address to an individual. Further, humans are incidental to our experimentation as our interest is in the assignment of wireless device layer-2 MAC addresses, or "what." Again, we have no way to map MAC addresses to individuals.

Finally, in consideration of beneficence and respect for persons, our work presents no expectation of harm, while the concomitant opportunity for network measurement and security provides a societal benefit. Our experiment was therefore determined to not be human subject research.

3.2 Building the Database

Algorithm 1 provides high-level pseudocode of our procedure to build a database that maps MAC addresses to various device models (along with available meta-data). For 802.11 management frames, we find those that contain WPS.

Table 3: Passively Collected Corpus Statistics

Frame	Count	%
Management		
- Probe Requests	67,086,700	3.25
- Probe Responses	134,639,147	6.53
- Beacons	1,051,269,586	50.98
- w/ WPS	29,121,890	†1.41
Data		
- mDNS w/dns.txt packets	806,954	0.04
- mDNS packets	2,503,800	0.12
- Unencrypted data packets (not mDNS)	119,328,932	5.79
- Encrypted	100,191,115	4.86
Other		
- Control & Unused Management Frames	586,406,377	28.4
Total	2,062,232,611	100.00

†WPS frames are an inclusive subset of listed Management Frames

We treat locally-assigned MAC addresses specially, as described in §3.2.1. Otherwise, we store the source MAC in a database with the frame's advertised WPS `manufacturer`, `model_name`, `model_number`, and `device_name` field values. Additionally, we store the frame's advertised `primary_device_type.category` and `subcategory` fields. A full enumeration of WPS device data is available in [29].

If WPS information is not available, we utilize mDNS. We parse mDNS packets for model identification key-value pairs (§2.3). If successful, we insert the source address and model string into the database, along with the device's common name derived via public resources [14, 21]. Devices are in a connected state when implementing mDNS, and as such are only observed using globally unique addresses.

mDNS, due to its inherent nature as a zero configuration Local Area Network (LAN) protocol requires the device to be in an authenticated and associated state. Hence, the mDNS data we collect is limited to those devices that were connected to an open 802.11 network. Our collection of mDNS is thus considerably smaller, allowing inference of only 10,525 devices as compared to 276,000 for WPS.

3.2.1 Locally Assigned MACs

Many devices, at one point or another, will use a locally assigned MAC address for P2P communication, hotspot mode, VLANs, or MAC address privacy.

• Privacy: We observe devices randomizing their MAC addresses when not associated with a WiFi network – the Motorola Nexus 6 and Huawei Nexus 6P, for example, switch to a predetermined local OUI while randomizing the lower three bytes. Regardless of the implementation details, many devices continue to send probe requests containing WPS information using this random, locally assigned MAC address. Using the method described in [27], we obtain the globally unique MAC via a pre-computed lookup table of the device's Universally Unique IDentifier-Enrollee (UUID-E), found in the WPS `uuid_e` field.

• P2P: We identify devices offering P2P capabilities via beacon frames that contain special extension tags (9 for WiFi-Direct and 10 for WiFi-Display [29]). We gather the source address of the device, the `P2P.device_id` (the locally assigned P2P MAC), the vendor extension tags, and Service Set IDentifier (SSID) advertised in the beacon frame. The SSID, while oftentimes user-configurable, generally defaults to a string identifying the manufacturer or model of the device to aid the user in discovering their WiFi-Direct devices. Last, we find the advertised `device_name`, a field that generally contains descriptive device information (a TV may advertise its model number, while a Roku media device provides a descriptive string).

Table 4: WPS-Based Device Category

Device Category	Total Number
PC	6803
Notebook	11
Amazon Tablets	2240
Input Device - Mouse	1
Printer	2501
Camera	596
Access Point	219796
Television/Display	326
Multimedia Device	287
Smart phone	36361
Audio Devices - Music Player	1
Gaming Systems	101

• Apple hotspots: Apple APs and devices acting as hotspots can be identified by the use of a standardized Vendor Specific `wlan_mgt.tag.oui` field (`0x0017F2` for Apple). The `tag.oui.type` subfield is used to differentiate between traditional APs (e.g., AirPort models) and client devices operating in hotspot mode, which advertise service using a locally assigned MAC address. Although beacon and probe frames from these devices do not contain model information, the OUI of the global MAC address is transmitted in a subfield of the Vendor Specific OUI field. We replace the locally assigned OUI used in hotspot mode with the derived OUI in order to identify the device's actual global MAC address.

3.3 Predicting Models from MACs

Using 802.11 management frames and unencrypted mDNS packets, we build a database that maps MAC addresses to a device manufacturer and model. To make an inference for an unknown device, we simply query the database for all known results that match at least the OUI of the device in question. We then perform a lexicographical comparison of the results to find the closest matching manufacturer and device. Further, we show the absolute distance between the two MAC addresses is a useful measure (§4.4). While simple, the size and coverage of our database allows us to make accurate inferences as we show next.

4. RESULTS AND ANALYSIS

In this section, we first present a broad overview of our 802.11 corpus and then analyze the allocation of MAC addresses by vendors, highlighting common practices and exceptions. Next, we validate our inference methodology against both a ground-truth set of devices as well as a third-party public data capture. We conclude with a 5-fold cross-validation test, using our own dataset to evaluate the effectiveness of our closest match methodology.

4.1 802.11 Corpus Statistics

As summarized in Table 3, our dataset contains over two billion 802.11 frames, of which approximately half are beacons. Probe requests and responses together comprise approximately 200M frames, of which approximately 15% contain WPS data. Considering only globally-unique MAC addresses, we observe 49,428 total client MAC addresses (distinguished by the device sending a probe request) using WPS, and 833,670 clients not implementing WPS. Among APs (or client devices acting as APs, e.g., hotspots), sending probe response or beacon frames we observe 227,428 distinct

Table 5: Top 10 Manufacturers - Clients
Indicates Strong/Weak WPS Inference Capability

WPS	Count	%	non-WPS	Count	%
LGE	11,184	22.60	Apple	231,214	44.36
Ralink	4,279	8.64	Samsung	48,617	9.33
Motorola	3,260	6.58	Murata	48,246	9.26
HTC	3,256	6.57	Intel	25,734	4.95
Prosoft	2,234	4.50	HP	15,287	2.94
Amazon	2,222	4.49	Microsoft	13,949	2.68
Huawei	1,905	3.83	Ezurio	12,385	2.38
Asus	1,659	3.34	Epson	6,839	1.32
ZTE	1,619	3.25	Lexmark	5,289	1.01
Alco	1,036	2.10	Sonos	4,542	.09
Other	16,859	34.10	Other	109,271	20.96

Table 6: Top 10 Manufacturers - APs
Indicates Strong/Weak WPS Inference Capability

WPS	Count	%	non-WPS	Count	%
Netgear	58,302	25.64	Cisco	73,144	12.30
Cisco	31,978	14.06	Ericsson	46,110	7.76
Linksys	22,440	9.87	Apple	40,105	6.75
Technicolor	19,295	8.48	Actiontec	39,350	6.62
Belkin	12,896	5.67	Ruckus	33,858	5.70
Arris	7,913	3.48	HP	27,600	4.64
ASUSTek	7,770	3.42	Aruba	23,429	3.94
Actiontec	7,163	3.15	Ubiquiti	18,126	3.05
Dlink	5,729	2.52	Cisco-Linksys	15,778	2.65
Broadcom	5,615	2.47	Mitsumi	12,523	2.10
Other	48,327	21.24	Other	264,483	44.49

MAC addresses whose frames include WPS data, and 1.79M without. Interestingly, some devices were observed including WPS data in some management frames, but at other times, not. 5,711 distinct client MAC addresses transmitted at least one management frame with WPS fields and at least one without; 204,353 unique AP MAC addresses fell into the same category. This behavior is common in APs that use WPS in probe responses, but not beacons.

WPS contains fields for the manufacturer to specify the general device category (e.g., PC, camera, etc.), allowing us to roughly characterize the population as summarized in Table 4. However, the device category advertised in WPS fields can be unintuitive or incorrect. For example, the Microsoft Xbox One gaming system sends "Ralink" in the corresponding manufacturer WPS field (the manufacturer of the chipset) and a PC device type. Similarly, Roku devices advertise as APs rather than multimedia devices.

We collect over 222M data frames, of which ∼45% are encrypted. We ignore non-mDNS and encrypted frames per our IRB agreement. 32% of the mDNS packets contain the `dns.txt` field, allowing us to identify the models of 9,849 Apple, 184 BlackBerry, and 417 Android devices.

Tables 5 and 6 reveal that Apple is the most prevalent client (∼44%) and third most prevalent AP (∼7%) manufacturer that does not implement WPS. We obtain the non-WPS statistics simply based on the OUI [13] for the set of management frames where no WPS data is included. We therefore use mDNS features to characterize Apple and other devices that do not use WPS.

Locally assigned MAC addresses are observed in over 166M frames (8%). Because locally assigned MAC addresses are used for P2P and privacy reasons (§3.2.1), it is difficult to ascertain exactly how many distinct devices transmitted these frames. Instead, we first consider devices using randomized local MACs that include a WPS UUID-E. From UUID-E identifiers that we can reverse using pre-computed tables, we find that the majority fall into seven distinct manufacturers: Motorola (502), Huawei (460), Samsung (259), Sony (91), HTC (71), Blackberry (36), and MediaTek (25). The most commonly observed models are the Motorola Nexus 6 (490), Huawei 6P (460), HTC Nexus 9 (71), Sony Xperia Z5 (69), Samsung S5 (38), BlackBerry Priv (36), and the Samsung Galaxy Note 4 (26).

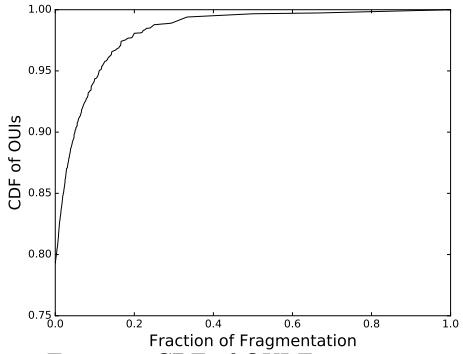

Figure 1: CDF of OUI Fragmentation

Next, we inspect the WPS fields among P2P frames. We observe 5,182 unique WiFi-Direct enabled devices, with 2,176 of those dual enabled for WiFi-Display. The majority of these devices are various Roku streaming media players (2,260) and Amazon Fire TV (148). A variety of HP, Samsung, and Epson printer models span 1,659 devices, with 302 Vizio and 282 Sony Bravia televisions.

Last, we observe 2,609 unique iPad and iPhone devices operating as hotspots. We calculate their global MAC using the technique described in §3.2.1. Two distinct bins for the offset values are present: an offset of either `0x02` or `0x22` in the first byte of the MAC address.

4.2 MAC Address Allocation

We next seek to characterize MAC allocation strategies employed in practice. There is no general pattern between manufacturers; some assign the entire OUI to only one model, while others assign smaller ranges to dozens of distinct models. The size and number of distinct ranges assigned to a model also follows no general rule. We highlight several exemplar manufacturers here.

We observe 2,956 unique OUIs. Inspection of the WPS data reveals that the 2,956 OUI contain ∼5,000 OUI to manufacturer pairings, and ∼10,000 OUI-model. This increase in pairings results from cases where the OUI is owned by the chipset manufacturer, further highlighting the value of fine-grained inference. Using mDNS, we see only Apple products within the OUI space allocated to Apple. We observe 352 distinct Apple OUIs and 1,028 unique OUI to model pairs.

In order to visualize MAC address allocation within an OUI, we plot occurrences of observed devices with a given OUI in different colors by model on an x-y plane, where the y-axis corresponds with the fourth byte of the MAC, and the x-axis with the fifth byte. To highlight the density of the MAC addresses we observe, we alter the gradient between each pair of sample points, such that the midpoint between two observed MAC addresses has minimum alpha value. We normalize the gradients to the largest distance between any two MAC addresses with the same model information; in this way, the largest distance between any two MAC addresses will appear white, indicating the lowest density of observation. This coloring also provides a visual indication of confidence in the inferred intervals – the most intense-colored portions indicate regions in which we have

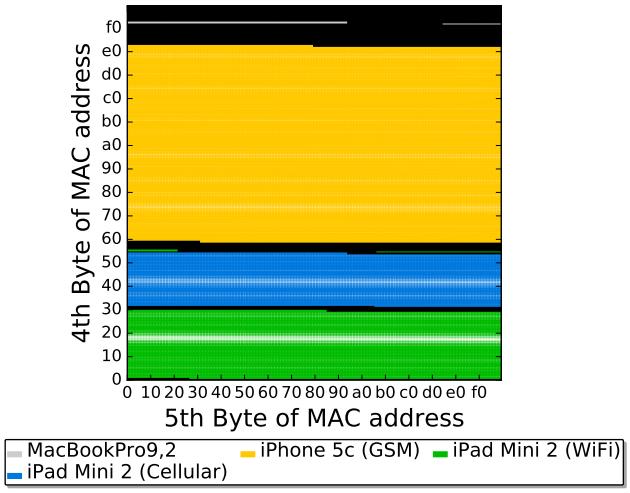

Figure 2: Observed Models in `24:A2:E1` (Apple)

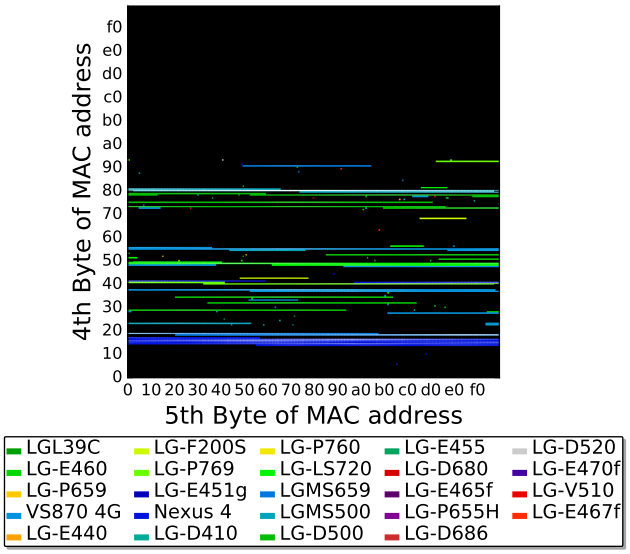

Figure 3: Observed Models in `8C:3A:E3` (LGE)

the highest confidence that these ranges of the OUI space will be of that particular model; conversely, in the whitest section of a contiguous block of MAC address to model observations, we have relatively lower confidence that a MAC address observed will correspond to the same model. Space within the OUI for which we have no observations or inferences based on a series of MAC addresses associated with the same model appear black.

In general, we find that MAC address allocation is non-random across vendors and OUIs. To quantify the amount of randomness present in model assignment of MAC addresses, we calculate the amount of "fragmentation" present in each OUI (Figure 1). We calculate fragmentation as follows: first, we find the number of MAC address triplets in lexicographical order in which the first and third MAC address correspond to the same model, while the second MAC is not. We then divide this count by the total number of triplets. Very little fragmentation appears in our database – approximately 80% of WPS and Apple mDNS OUIs have no fragmented MAC address-model triplets.

4.2.1 *Example: Apple*

Our first example is Apple's `24:A2:E1` OUI which contains four models: two models of the iPad Mini 2 (cellular and WiFi-only versions), the iPhone 5c, and a 13" MacBook Pro. Figure 2 graphically displays our inferred ranges for these models within the OUI, with black intervals indicating portions with no observations. All OUI plots in this work (like Figure 2) are in color should be viewed with color.

Based on the start and end of each contiguous block of MAC address observations, we infer that the largest part of the OUI is dedicated to the iPhone 5c, with a block consisting of over 8.7 million MAC addresses (52% of the OUI). The 3.06M MAC addresses allocated to the WiFi version of the iPad Mini 2 are spread among a contiguous block of 3.03M addresses and a much smaller block of 31K addresses in a different portion of the OUI. ∼2.4M addresses are allocated to the cellular version of the iPad Mini 2, and a small range (∼50K) is devoted to the "MacBookPro 9,2" (a mid-2012 13" laptop). Interestingly, the iPad Mini 2 and iPhone 5c were released in late 2013, over a year after the release of this MacBook Pro, illustrating that allocations evolve over time. Based on the MAC address-model pairs

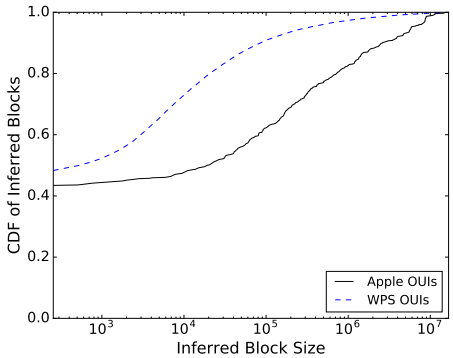

Figure 4: CDFs of Inferred Block Sizes for WPS-Using Manufacturers and Apple

in our database, we find that we can make no inference for ∼15% of the `24:A2:E1` OUI, indicated by the black portions of Figure 2.

4.2.2 *Example: LG Electronics*

In stark contrast to our inferred allocation of the Apple OUI is the `8C:3A:E3` OUI, registered to LG Electronics. LG manufactures many 802.11-enabled devices; in this particular OUI we observe twenty-one distinct models of smartphones. Our inference of the MAC address ranges indicated spans only 9% of the OUI – large continuous blocks of models are rarely observed; those that do, span only several bytes in the fourth byte of the MAC address. Many observed data points alternate models of smartphone, thereby allowing us to make no inference beyond a single 256-MAC address pixel as to ranges assigned to particular models. For instance, the LGL39C, a prepaid device sold by Tracfone, appears four times in our database, but never without another model of phone between two data points. We therefore are confident only in the particular five-byte point in which the LGL39C appears. It bears noting that some OUI of LG phones contain multiple models within the same 256-MAC address, five-byte pixel as well, implying that some manufacturers allocate multiple devices within a single five-byte range of MAC addresses. Such micro-allocations sig-

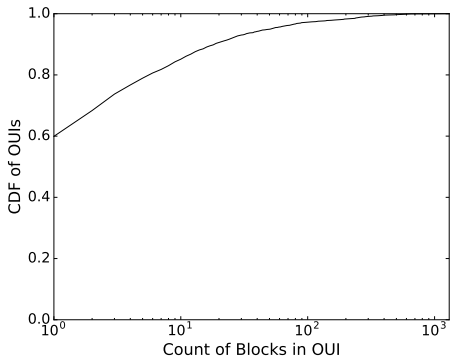

Figure 5: CDF of Number of Blocks per OUI (all manufacturers)

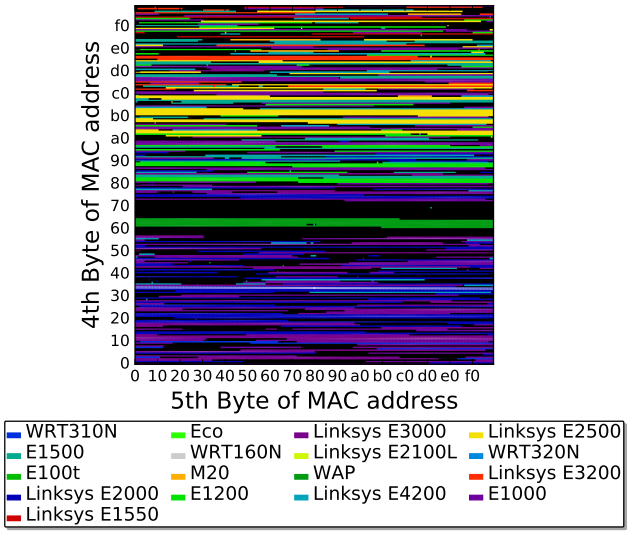

Figure 6: Observed Models in `C0:C1:C0` (Cisco)

nificantly increase the difficulty for a passive observer to accurately infer the ranges of MAC addresses assigned to individual models. In general, we observe more fine-grained MAC address ranges for manufacturers using WPS (like LG) than we do for Apple, from whom we gather model information through mDNS. Figure 4 illustrates the relative difference in inferred block sizes, comparing all 58 Apple OUIs for which we have data against the top 58 OUIs in which we derive model-level detail from WPS. In Figure 5, we plot the distribution of number of inferred blocks per OUI. While ∼60% of OUIs have only one inferred block, a nontrivial percentage (∼10%) have 20 or more.

4.2.3 *Example: Cisco*

Allocations of contiguous blocks to distinct device models are present among AP manufacturers as well. Figure 6 shows the inferred allocation of the `C0:C1:C0` OUI owned by Cisco, with 17 models of APs. Visually, our plot indicates that assignment of addresses is performed more granularly by Cisco than Apple. As opposed to the five distinct blocks in §4.2.1, we find 248 distinct contiguous ranges dispersed throughout the OUI. The Linksys E1000 appears most frequently, with 54 distinct ranges of addresses that transmit management packets with WPS. Further, the blocks associated with the E1000 also make up the largest allocation to any single device (2.4M addresses). Two devices appear in

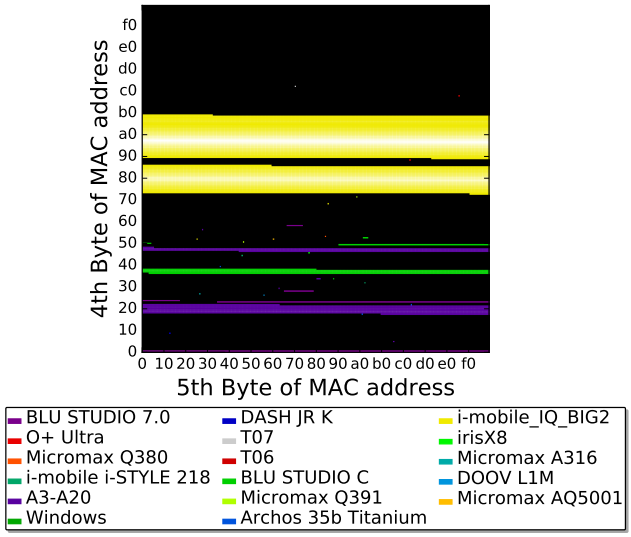

Figure 7: Observed Models in `90:21:81` (Shanghai Huaqin)

a single x-y grid square – the Cisco E100t, a network card, and Linksys E2100L, a wireless router, giving these two devices the smallest inferred ranges within the OUI. Overall, we infer that about 60% of the OUI has been allocated, with a mean of ∼41K addresses per block.

4.2.4 *Example: Spanning OUIs*

To highlight the complexity and diversity of address allocation policies, we present two interesting examples: manufacturers that split MAC address ranges assigned to a single model across multiple OUIs, and OUI that contain devices produced by multiple manufacturers.

To illustrate the assignment of the same device model throughout the ranges of several OUI, we examine another Apple-owned OUI that contains iPhone 5c devices, as in §4.2.1. The 2^{24} addresses in `0C:3E:9F` are split between the GSM versions of the iPhone 5c and 5s, with 33% and 53% of the OUI respectively allocated to each. iPhone 5c GSM devices are found in ten Apple OUIs; including the global variant, twelve OUIs contain some version of the iPhone 5c. This model is by no means unique in its distribution across multiple OUIs – our collection has discovered iPhone 6 Plus devices in three OUIs – nor is allocating a single device to several OUIs unique to Apple. For example, we observe Huawei's Nexus 6P in nine distinct OUIs. We speculate that the rationale behind dividing a single device to multiple OUIs may be one of efficiency (that is, maximizing utilization of previously-purchased OUI space) or possibly indicative of a logical assignment of an OUI to discrete manufacturing locations.

Further, we present an illuminating counterexample to the conventional wisdom that an OUI is sufficient to identify the manufacturer of the device. Figure 7 is a visualization of the `90:21:81` OUI where we observe seven distinct manufacturers: Acer, Archos, BLU Mobile, i-Mobile, LAVA, Micromax, and Oplus with twenty-eight address ranges assigned to twelve device models. Because we have been able to infer address blocks for only ∼23% of the OUI, it is likely that more models and manufacturers occupy the OUI space. This finding is likely attributable to the OUI owner (Shanghai Huaqin Telecom) producing 802.11 chipsets for many

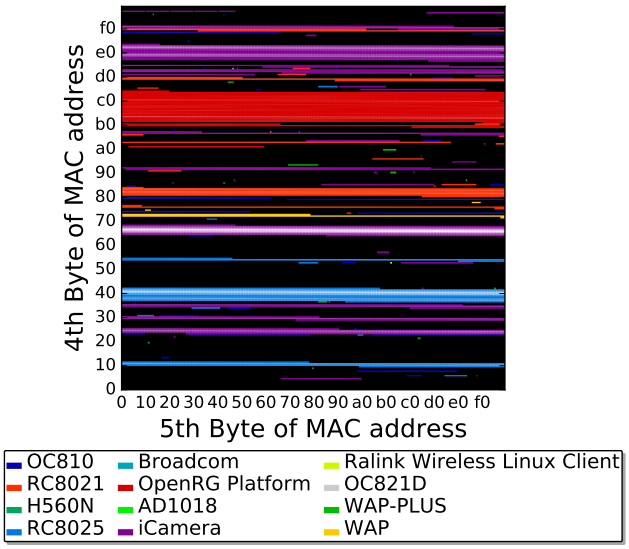

Figure 8: Observed Models in `00:0E:8F` (Sercomm Corp.)

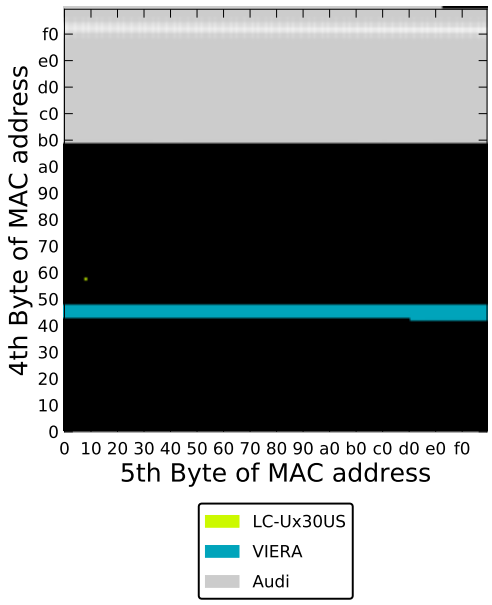

Figure 9: Observed Models in `48:A9:D2` (Wistron Neweb)

manufacturers of low-cost devices. Whereas fingerprinting of devices based on the OUI would identify a single manufacturer, our technique finds granular discrete ranges for unique manufacturers and models within this OUI.

4.2.5 *Example: IoT*

We also apply our OUI allocation inference to IoT devices. Figure 8 highlights an interesting result, the `00:0E:8F` OUI registered to Sercomm Corp. While portions of the OUI are allocated for APs, we observe a wireless repeater (ZTE H560N) and five types of 802.11-enabled cameras, the Sensormatic OC810, OC821D, RC8025, and RC8021 models, and iCamera 1000. The iCamera, in particular, appears 121 times in twenty-seven discrete blocks. We note that a thief conducting wireless reconnaissance of a potential target could easily determine whether or not one of these cameras was present using our fine-grained fingerprinting.

Next, we show two use cases where using identifiers passed through WPS or mDNS can be augmented with other, more commonly used methods of fingerprinting devices. Figure 9 shows the `48:A9:D2` OUI of Wistron Neweb Corp. Evaluation of the byte allocation using WPS reveals four devices, one Sharp and three Panasonic Vierra televisions; there are no results from the analysis of mDNS in this OUI. To more thoroughly evaluate the allocation structure, we augment our analysis by manually inspecting beacon frames from devices in this OUI. We find a common and consistent SSID syntax used by Audi vehicle WiFi systems. Furthermore, a large contiguous block is allocated to such vehicle-based WiFi systems.

These results highlight several observations; i) the ability to fingerprint diverse IoT device allocations; ii) structure can be augmented with inspection of common SSIDs; and therefore iii) multiple methods of identifying byte allocation structure exist and are complimentary.

4.3 Validation on CRAWDAD Sapienza Data

To evaluate our technique, we validate against a third-party corpus of publicly available probe requests in CRAWDAD from Sapienza [6]. This dataset consists of approximately 11M probes requests on behalf of 160,000 unique devices. The Sapienza CRAWDAD corpus differs from ours in the location and time period during which it was obtained – our data is primarily from the Eastern Seaboard of the United States from 2015-2016, while the Sapienza data is from three months in 2013 throughout Italy.

The Sapienza corpus has anonymized the 802.11 probe MAC addresses, SSIDs, and other identifiers. To obtain device MAC addresses, we therefore take the subset of probe requests containing WPS fields that have both a distinguishing manufacturer and model field, as well as a UUID-E that we can invert using the technique described in [27][1]. Of the 2,674 unique UUID-Es, we map 1,832 back to the true global MAC addresses. Finally, we remove 86 addresses with only "SAMSUNG ELECTRONICS" as the manufacturer and "SAMSUNG MOBILE" as the model, as they provide no detail on the precise model or device. Thus, we perform validation on a test set of 1,746 real MAC addresses from 63 distinct OUIs with corresponding manufacturer and model labels from the WPS probe advertisements.

We use the technique of §3.3 to query for each of the MAC addresses in the Sapienza corpus, and compare our inferred manufacturer and model against the manufacturer and model in the corpus' WPS frame. We find that our inferences correctly predict the correct manufacturer and model for 1,419 of the 1,746 addresses, an 81.3% accuracy.

We make three observations regarding the ∼19% of incorrect predictions. First, our database returns no results (that is, cannot match at least the first three bytes of the queried MAC) only four times. Three of these database "misses" are attributable to Samsung Galaxy Tab tablets, a device that we have observed in the wild infrequently. The fourth is a mobile phone sold in Korea, the Pantech Vega R3 (IM-A850K), which we have not observed in our corpus. Similarly, other devices that we mispredict are observed rarely in our collection, or not at all. For example, the mobile phone manufacturer Pulid appears twice in our dataset, but in different OUIs from the Pulid phone present

[1]We have notified the corpus authors of the requirement to also anonymize UUID-E.

Table 7: Validation on Ground-Truth Devices

Device	Precision	Recall	F-score
Apple			
- iPhone (iOS 7.0-)	.000	.000	0
- iPhone (iOS 8.0+)	.909	.909	.909
- iPad/iPod (iOS 8.0+)	.857	.900	.877
- All iOS 8.0+ Devices	.892	.906	.898
- OS X	.771	1.00	.870
- Apple TV	.750	1.00	.857
- iOS 8.0+ and OS X	.850	.934	.890
- All	.715	.838	.772
Samsung			
- Galaxy S4 and prior	.684	.892	.774
- Galaxy S5 to current	.475	.863	.613
- Galaxy Tablets	.250	.071	.110
- All	.598	.761	.670

in the Sapienza dataset, while we never observe the manufacturer "Jiayu." Finally, many of the incorrect predictions are, in fact, temporally "close," meaning that the release date of the device closely matches our prediction. For instance, the Sapienza corpus contains a large number of Samsung Galaxy S III smartphones under various carrier-defined model names (e.g., GT-I9300). While our system correctly guesses the model for the majority of Samsung Galaxy S III devices, the most common incorrect predict is a variant of the Samsung Galaxy Note II. Given that these two phones were released three months apart, and that the Galaxy Note II is based on the hardware design of the Galaxy S III, even this incorrect inference provides useful context about the device.

4.4 Validation on Known Device Dataset

As a second method of validation, we evaluate our fingerprinting against a set of known, ground-truth devices. We collect 140 Apple and 139 Samsung devices, and manually obtain layer-2 addresses from their settings menus. The devices range across device types (phones, TVs, tablets), life-cycles (2007-2016), and operating systems (iOS 1-9.3, Android OS 1.5-6.0). While our set of known devices is significantly smaller than in the CRAWDAD Sapienza corpus, we definitively know ground-truth without relying on WPS-based device types. As such, the Apple devices specifically evaluate the power of the mDNS derived allocations in our database.

Again, using our technique in §3.3, we predict each device's manufacturer and model and compare against the true manufacturer and model. Additionally, we record the number of matching bytes for the lexicographically closest match in our database, along with the protocol (WPS or mDNS) that provides the closest match.

Table 7 provides our inference precision, recall, and F-score (the harmonic mean of the precision and recall). Precision evaluates when we provide a prediction is that prediction correct, where recall factors in the ability for our inference model to make a prediction. In cases where we have no three byte match we fail to make any inference.

We do not correctly identify any of the iOS 7.0 devices. The distinction between iOS 7.0 and 8.0+ is significant, but easily explained. The iOS model derivation process relies on the inclusion of a `dns.txt` key-value pair within the mDNS packet, which is only sent by models running iOS 8.0 or later.

The majority of devices that are either incorrectly identified or produce no match even at the base OUI level are simply due to a lack of observations in our passive data collection. For example, we observe no iPad Pros in our collection, and therefore currently have no insight into the allocation structure – Table 8 depicts this scenario.

Table 8: Ground-Truth Inference – Offset Comparison

Device	Best Guess	Byte Match	Offset
iPad 2	iPad 2	3	2
iPad 2	iPad 2	3	31
iPad 2	iPad 2	3	11
iPad 3	iPad 3	3	2
iPad Air	iPad Air	3	2
iPad Air	iPad Air	3	1
iPad Air 2	iPad Air 2	3	7
iPad Air 2	iPad Air 2	3	1
iPad Air 2	iPad Air 2	4	18
iPad Air 2	iPad Air 2	3	6
iPad Mini	iPad Mini	3	5
iPad Mini	iPad Mini	3	1
iPad Mini 3	iPad Mini 3	4	58
iPad Mini 4	iPhone 6	3	104
iPad Mini 4	iPhone 6	3	186
iPad Mini 4	iPhone 6	3	94
iPad Pro			
iPad Pro			

For an arbitrary MAC address queried against our database, we define the match *offset* as the absolute difference of the $(n+1)^{th}$ byte of the test MAC address and the closest database match, where both MAC addresses match through n bytes ($n \geq 3$). We observe that the accuracy of the result increases as the offset of the match decreases. For example, in Table 8, the three iPad Mini 4 devices have a three byte (OUI) match with corresponding high offset values (in decimal: 104, 186, 94). The highest offset value of any successfully inferred iPad model is 31.

We obtain a lower overall accuracy as compared to our Sapienza validation test which, interestingly, contains a large percentage (∼60%) of Samsung devices. For Samsung models common to the time period of the Sapienza CRAWDAD collection (Galaxy S4 and earlier), we achieve similar fingerprinting performance. We posit that accuracy difference is due to several recent trends we observe in the use of WPS and mDNS by Samsung devices. For older models (circa 2013), our dataset contains 630 WPS Samsung devices, and only 30 Galaxy S5 or newer. An additional 250 Samsung devices transmit the WPS fields only while using a locally assigned MAC address.

As a corollary, we examine the benefit associated with increased data collection for our methodology. We plot our successes and failures for the Sapienza CRAWDAD dataset and our own Apple and Samsung devices according to the density of the inferred block in which they fall. That is, if a test MAC address falls within an inferred block of size 2048 and 64 instances of the block's model within this range, we say the block has density 0.03125. On the other hand, if a test MAC address falls outside of a block, we say the inferred block has 0 density.

Figure 10 displays the CDFs of the block densities for our correct and incorrect model inferences. Only 45% of Sapienza correct inferences fall outside of a model block from our database, but are closer to the edge of a block of the correct model than an incorrect model. Conversely, 55% of correct inferences fall inside a block of nontrivial density. Of the Sapienza CRAWDAD MAC addresses for which we made incorrect model inferences, 85% fall outside of any block and hence have a block density of 0. Figure 10 also shows the CDF of the block densities of our correct Apple

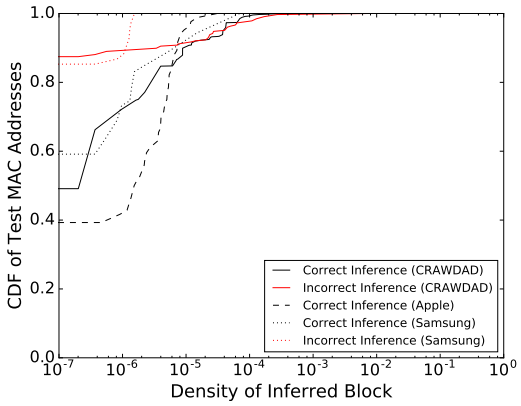

Figure 10: Relationship between observation density and inference ability

(mDNS) and Samsung (both WPS and mDNS) inferences for which ground truth is known. Less than 40% of correct Apple inferences fall outside a block, while the majority have densities between 2×10^{-6} and 8×10^{-6}. We do not plot our incorrect Apple inferences for our known devices – only one incorrect inference was made with a nonzero density. Overall, our results demonstrate that when a MAC address falls inside a block, a higher density indicates an increased likelihood that our guess will be correct.

4.5 Cross-Validation

Last, we conduct a 5-fold cross-validation using our own dataset to evaluate the effectiveness of our closest match methodology. We partition both our WPS and Apple mDNS collections into five randomly chosen sets of equal size. For all MAC addresses in each set, we find the closest matching MAC address in any of the other four sets by both simple distance (interpreting MAC addresses as 48-bit integers) and lexicographical distance. Each set is used once as validation (test) data against the remaining four sets (training); we then compute the arithmetic mean of the accuracies across the five folds.

The WPS 5-fold cross-validation yields an average accuracy of 90.95% using lexicographical distance and 91.16% with simple distance. The simple distance closest match strategy outperforms closest lexicographical distance, though only by 0.2%, suggesting that some manufacturers may prefer allocating MAC address blocks across 4-byte boundaries rather than assigning "prefixes" to individual models. This cross-validation accuracy represents an ∼10% improvement over the accuracy we obtain when testing our model against the older and geographically distant CRAWDAD Sapienza dataset in §4.3 – and demonstrates the utility in obtaining as many disparate training samples as possible.

Our Apple mDNS 5-fold cross-validation also yields an improvement over the performance of our model using our set of ground-truth devices. With the simple distance closest match metric, we correctly identify the iOS 8.0+ or OS X (iOS 7 and below do not include model-identifying information in mDNS messages) device with an average accuracy of 88.16% across all five validation folds. The lexicographical closest matching method is accurate 88.20% of the time. Both methods represent a ∼3% improvement over our validation using the set of Apple devices for which we were able to obtain ground-truth.

In summary, our cross-validation demonstrates an approximately 90% success rate in correctly identifying manufacturer and model based on a MAC address alone. Cross-validation minimizes the effect of evaluating our model against non-US devices (a weakness of our validation in §4.3), and functions as a more robust device validation set than is possible for individuals to collect (a weakness of §4.4.)

5. CONCLUSIONS

Our data suggests that MAC address assignment policy is nonrandom – vendors of smartphones and tablets, APs, and IoT devices allocate contiguous blocks from their OUIs to individual device models. However, we find no "standard" MAC address allocation strategy. While Apple OUIs are divided into several large chunks for a small number of devices, other manufacturers (e.g. LG and Cisco) allocate small blocks to a larger number of device models.

Our rich dataset, consisting of over two billion 802.11 frames and approximately 3,000 OUIs, allows us to make device model granular predictions for unknown MAC addresses. Not only do our inferences improve the granularity of MAC-based fingerprinting, we also show that allocation policies are sufficiently varied and complex as to cause simple fingerprinting techniques to be inaccurate. For instance, we discover devices that span multiple OUIs as well single OUIs that contain devices from multiple vendors – as many as seven different manufacturers (§4.2.4).

To validate our ability to form model-level predictions, we test our inferences against a ground-truth set of 279 devices where we achieve an F-score of 0.85 to 0.91 for Apple devices (depending on specific model) and an F-score of 0.61 and 0.77 for Samsung devices. We then evaluate our inferences against a public corpus of 802.11 probe requests. In this third-party dataset, collected on a different continent than our own corpus, we achieve 81% accuracy, with most of the errors being relatively close, i.e. due to re-branded carrier-defined model names.

5.1 Future Work

We leave four items for further investigation. First, during the course of our 802.11 data collection, we observed several link-layer discovery protocols leaking manufacturers and models. These protocols, including Cisco Discovery Protocol (CDP), MicroTik Network Discovery Protocol (MNDP) and others, could provide the basis for an analogous MAC to device model mapping for wired infrastructure.

Second, we hypothesize that MAC addresses can provide a geographic indication of where the wireless device was purchased or originated. With our continued data collection, we hope to provide insight into the locality of particular ranges of addresses within an OUI. Such a "geo-distribution" of MAC addresses and models could provide valuable analytics to industry and government, providing the ability to infer what region of the world devices (and their owners, by proxy) are from. We also conjecture that MAC addresses are likely assigned in a sequential manner (or semi-sequential, accounting for Bluetooth MAC address assignment) *within* device model blocks, indicating a relative manufacture date.

Our findings naturally beg the question of how one might evade model-level fingerprinting. Not only can the MAC address be spoofed, provided that the WPS UUID-E is calculated to match, an adversary might attempt to poison our mapping database by advertising inaccurate data in WPS

management frames or mDNS packets. Future work should consider ways in which to make the inferred address mappings more robust to such attacks.

Finally, we encourage manufacturers to adopt a more fine-grained allocation of MAC addresses to particular devices. As we see in §4.2.2 and §4.2.3, the smaller the allocated ranges of MAC addresses are to individual models, the more complex and difficult it is to infer the structure of an OUI (and hence, decrease the inference power of our database).

Acknowledgments

We thank Adam Aviv, Mark Gondree, Travis Mayberry, and Justin Rohrer for early feedback. This work supported in part by NSF grant CNS-1213155. Views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of the U.S. government.

6. REFERENCES

[1] Deviceatlas. May 2016. http://deviceatlas.com.

[2] Fingerbank. Sept. 2016. https://fingerbank.inverse.ca.

[3] Udger. May 2016. http://udger.com.

[4] Wireless Universal Resource File (WURFL), May 2016. http://wurfl.sourceforge.io.

[5] D. E. 3rd and J. Abley. IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters. RFC 7042 (Best Current Practice), Oct. 2013.

[6] M. V. Barbera, A. Epasto, A. Mei, S. Kosta, V. C. Perta, and J. Stefa. CRAWDAD Dataset Sapienza/Probe-Requests. http://crawdad.org/sapienza/probe-requests/20130910, Sept. 2013.

[7] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*, 2011.

[8] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying Unique Devices through Wireless Fingerprinting. In *Proc. ACM WiSec*, 2008.

[9] C. V. Dr. Charlie Miller. Remote Exploitation of an Unaltered Passenger Vehicle, Aug. 2015. http://illmatics.com/Remote%20Car%20Hacking.pdf.

[10] D. Evans. The Internet of Things: How the Next Evolution of the Internet is Changing Everything, Apr. 2011. http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[11] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *Proc. 15th USENIX Security Symposium*, 2006.

[12] T. Hupperich, D. Maiorca, M. Kührer, T. Holz, and G. Giacinto. On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms? In *Proc of Annual Computer Security Applications Conference*, 2015.

[13] IEEE. OUI Public Listing. http://standards.ieee.org/develop/regauth/oui/oui.txt.

[14] iOS Device Database Wiki, May 2016. http://www.enterpriseios.com/wiki/iOS_Devices.

[15] B. Konings, C. Bachmaier, F. Schaub, and M. Weber. Device Names in the Wild: Investigating Privacy Risks of Zero Configuration Networking. In *IEEE 14th International Conference on Mobile Data Management*, 2013.

[16] M. W. L. Schauer and P. Marcus. Estimating Crowd Densities and Pedestrian Flows using Wi-Fi and Bluetooth. In *in Proceedings of the International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2014.

[17] J. Lee. Using Guided Missiles in Drive-Bys. DEFCON-17, 2009. https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-egypt-guided_missiles_metasploit.pdf.

[18] Libelium. Meshlium Xtreme Technical Guide v5.9, May 2016. http://www.libelium.com/development/meshlium/documentation/meshlium-technical-guide.pdf.

[19] A. D. Luzio and J. S. A. Mei. Mind Your Probes: De-Anonymization of Large Crowds through Smartphone WiFi Probe Requests. In *IEEE INFOCOM*, 2016.

[20] J. Martin, D. Rhame, R. Beverly, and J. McEachen. Correlating GSM and 802.11 Hardware Identifiers. In *IEEE Military Communications Conference*, 2013.

[21] B. O. Models, May 2016. Chimeratool [Online]. Available: https://chimeratool.com/help/How-to-Unlock-OS10-Blackberry.

[22] J. Oliver. glasshole.sh, July 2014. https://julianoliver.com/output/log_2014-05-30_20-52.

[23] J. Oliver. Cyborgunplug, Mar. 2016. https://github.com/JulianOliver/CyborgUnplug.

[24] S. Chesire, M. Krochmal. DNS-Based Service Discovery. RFC 6763, Feb. 2013.

[25] L. Vaas. Nordstrom Tracking Customer Movement via Smartphones' WiFi Sniffing, 2013. https://nakedsecurity.sophos.com/2013/05/09/nordstrom-tracking-customer-smartphones-wifi-sniffing/.

[26] L. Vaas. Businesses are Building Shopper Profiles Based on Sniffing Phones' WiFi, 2014. https://nakedsecurity.sophos.com/2014/01/16/businesses-are-building-shopper-profiles-based-on-sniffing-phones-wifi/.

[27] M. Vanhoef, C. Matte, M. Cunche, L. Cardoso, and F. Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *ACM AsiaCCS*, 2016.

[28] R. H. Weber. Internet of Things–New Security and Privacy Challenges. *Computer Law & Security Review*, 26(1):23–30, 2010.

[29] Wi-Fi Alliance. Wi-Fi Peer-to-Peer Services (P2Ps) Technical Specification (for Wi-Fi Direct Services Certification). *Wi-Fi Alliance Document*, 2014.